TEACHERS, LEARNERS AND ORACLES

ACHILLES A. BEROS AND COLIN DE LA HIGUERA

Abstract. We exhibit a family of computably enumerable sets which can be learned within polynomial resource bounds given access only to a teacher, but which requires exponential resources to be learned given access only to a membership oracle. In general, we compare the families that can be learned with and without teachers and oracles for four measures of efficient learning.

1. Introduction

In this paper, we address the question of whether or not the presence of a teacher as a computational aide improves learning. A teacher is a computable machine that receives data and selects a subset of the data. In the models we consider, a teacher receives an enumeration for a target and passes its data selection to the learner – the learner does not have access to the original data. The first natural question is if there are families that are learnable with a teacher, but not learnable without. As will be obvious from the definitions presented in the next section, the answer is no: the learner can always perform an internal simulation of the learner-teacher interaction and output the result. The second question is whether a teacher can improve efficiency. For teacher models of learning, only the computational activity of the learner counts against the efficiency bound; the computational activity of the teacher is not counted. Heuristically, the question is whether there is benefit to pre-processing data. We will prove there can be an exponential improvement in efficiency. In fact, there are situations where access to a teacher is better than access to a membership oracle about the target.

Various forms of and questions related to teaching have arisen in learning theory over the last few decades. Work on the complexity of teaching families has given rise to the classical teaching dimension [7] and more recently the recursive teaching dimension [12, 4]. In [4], Zilles et al. establish deep and interesting connections between recursive teaching dimension, Vapnik-Chervonenkis dimension and sample compression schemes (see [5] for more about sample compression). Query learning has been a central topic in learning theory for even longer than teaching. Numerous papers have been written both on the abilities of machines equipped with oracles to learn [10, 1, 2] and on the properties of oracles that allow learning of certain target families [11, 8, 6, 9].

We add to the body of research on teaching and query learning by comparing the efficiency of the two learning modes.

2. Background

We will examine variants of Gold-style text learning of effectively describable sets of natural numbers. In particular, the target objects will be *computably enumerable sets*.

Definition 2.1. A set, S, is *computably enumerable (c.e.)* if there is a partial computable function, f, such that $S = \mathrm{dom}(f)$. A sequence of sets, $\{A_n\}_{n\in\mathbb{N}}$ is called *uniformly computably enumerable (u.c.e.)* if the set $\{\langle a, i\rangle : a \in A_i\}$ is c.e. We also call u.c.e. sequences of sets *indexed families* and call n an index for A_n. Note that in an indexed family a set may have multiple indices if the sequence $\{A_n\}_{n\in\mathbb{N}}$ has multiple instances of the same set. For notational convenience, we regard indexed familes both as sequences and as sets and write $A \in \mathcal{A}$ meaning $(\exists n)(A = A_n)$.

We now remind the reader of some standard notation and concepts as well as introducing some notation specific to this paper.

(1) ϕ denotes an acceptable universal Turing machine and hence, a partial computable function. $\phi_{e,s}(x)$ is the state or value of the function described by the program coded by $e \in \mathbb{N}$ after s computation stages on input x. If the program execution has terminated, we write $\phi_{e,s}(x)\downarrow$, otherwise we write $\phi_{e,s}(x)\uparrow$.
(2) W_e is the c.e. set coded by the program e as the domain of ϕ_e. $\{W_e\}_{e\in\mathbb{N}}$ is a u.c.e. sequence of sets and enumerates all the c.e. sets. We write $\mathcal{E}$ for the set of all c.e. sets.
(3) For $n \in \mathbb{N}$, $\langle x_0, x_1, \dots, x_n\rangle : \mathbb{N}^{n+1} \to \mathbb{N}$ is a polynomial-time computable encoding function such that $x_i \leq \langle x_0, x_1, \dots, x_n\rangle$ for all $i \leq n$. We also define a polynomial-time computable decoding function $(x)_n : \mathbb{N} \to \mathbb{N}^n$ which is the inverse function of encoding function $\langle x_0, x_1, \dots, x_{n-1}\rangle$. We define $A \otimes B = \{\langle a, b\rangle : (a \in A) \wedge (b \in B)\}$. We use $\otimes$ to partition $\mathbb{N}$ into an infinite number of infinite computable sets, $\mathbb{N}\otimes\{0\}, \mathbb{N}\otimes\{1\}, \dots$. Sets of this form are known as *columns*, whereby $\mathbb{N}\otimes\{i\}$ is the i^{th} column of $\mathbb{N}$. As a shorthand, we will represent the i^{th}-column of $\mathbb{N}$ with the symbol C_i and the i^{th}-column of $A \subseteq \mathbb{N}$ by $C_i(A)$. Associated with C_i, we define c_i to be a computable function such that $W_{c_i(x)} = W_x \cap C_i$.
(4) We write $(x_0, x_1, \dots, x_n)$ to denote the ordered tuple of elements (as opposed to the encoding of the ordered tuple, $\langle x_0, x_1, \dots, x_n\rangle$).
(5) We fix an encoding of polynomials as natural numbers and write p^* to denote the encoding of a polynomial p. The encoding is polynomial-time computable, as is the decoding, and maps onto $\mathbb{N}$.
(6) $\textsc{signedInt} : \mathbb{N} \to \mathbb{Z}$ is the computable bijection such that $\textsc{signedInt}(2n) = n$ and $\textsc{signedInt}(2n+1) = -(n+1)$.
(7) If a is a string or natural number, then a^i denotes the string which consists of a repeated i times.
(8) For function composition we use the notation $f \circ g$ where $(f \circ g)(x) = f(g(x))$.
(9) If $\sigma = a_0 \cdots a_n$ is a string, then $|\sigma| = n + 1$ is the length of the string, $\sigma(k) = a_k$ and $\mathrm{content}(\sigma) = \{\sigma(k) : k < |\sigma|\}$.
(10) An enumeration of a non-empty set A is an infinite sequence of elements of A such that every element of A appears in the sequence at least once. We regard an enumeration as a stream of bits with markers between individual elements. We will restrict our attention to non-empty sets. Consequently, we need not consider enumerations of the empty set.
(11) A learning machine (or learner) is a partial computable function that receives a string as input, may have access to oracle queries and outputs a natural number that is interpreted as a code for a set. The outputs are called hypotheses and the sequence of

hypotheses produced by a learner on initial segments of an enumeration is called the hypothesis stream. When measuring efficiency, we allow a learner to skip an element of an enumeration for some fixed computational cost.

(12) Given an interval $[0,n]$, where n is unknown, but bounded by a^m, n can be determined with $(m+1)^{a+1}$ or fewer oracle queries using the following algorithm. First, determine the least k_0 such that $a^{k_0+1} \notin [0,n]$. We will obtain k_0 after at most $m+1$ queries. Next, we repeat the process to determine the least k_1 such that $a^{k_0} + a^{k_1+1} \notin [a^{k_0}, n]$. By iterating this process at most $a+1$ times we find n. We call this an exponential query search algorithm.

We will consider learning models using combinations of three different data sources: enumeration, oracle and teacher. All of the models we consider are forms of TxtEx-learning, or learning in the limit. We begin with the definition of this fundamental learning model.

Definition 2.2. Let M be a computable learning machine, $\mathcal{F} = \{F_n\}_{n\in\mathbb{N}}$ an indexed family and $\{a_n\}_{n\in\mathbb{N}}$ an enumeration (text) of a set $F \in \mathcal{F}$.

(1) M TxtEx-identifies $\{a_n\}_{n\in\mathbb{N}}$ if

$$(\exists i)(\forall j)\big(M(a_0 \dots a_{i+j}) = M(a_0 \dots a_i) \wedge F_{M(a_0\dots a_i)} = F\big)$$

If only the first condition above is met, i.e., $(\exists i)(\forall j)(M(a_0...a_{j+1}) = M(a_0...a_i))$, then we say that M has *converged* on the enumeration $\{a_n\}_{n\in\mathbb{N}}$.

(2) M TxtEx-learns F if M TxtEx-identifies every enumeration of F.

(3) M TxtEx-learns $\mathcal{F}$ if M TxtEx-learns every $F \in \mathcal{F}$.

All of the models we examine in this paper are variants of TxtEx-learning. The parameters we will vary are linked to sources of information and the measurement of efficiency. We state definitions of these variants starting from an arbitrary learning model.

Definition 2.3. Let L-learning be an arbitrary learning model.

(1) We say that $\mathcal{F}$ is *L-learnable with a membership oracle* (denoted L[O]-learnable) if there is a learning machine, M, that L-learns $\mathcal{F}$ and has access to a membership oracle for the target it is learning. As membership oracles are the only oracles we will consider, we often simply refer to a membership oracle as an oracle.

(2) A function $T : 2^{<\mathbb{N}} \to 2^{<\mathbb{N}}$ is a *teacher* if it is a computable function, $T(\sigma)$ is a prefix of $T(\tau)$ whenever σ is a prefix of τ, and content$(T(\sigma)) \subseteq$ content(σ). We say that $\mathcal{F}$ is *L-learnable with a teacher* (denoted L[T]-learnable) if there is a learner-teacher pair (M,T) such that M L-identifies every enumeration of the form $T \circ f$, where f enumerates a member of $\mathcal{F}$.

(3) We say that $\mathcal{F}$ is *L-learnable with a teacher and a membership oracle* (denoted L[T,O]-learnable) if there is a learner-teacher pair, (M,T), such that M has access to a membership oracle, T has access to the query responses M receives, and M L-identifies every enumeration of the form $T \circ f$, where f enumerates a member of $\mathcal{F}$.

As is clear from the definition, the teacher serves to pre-process the text input before passing the elements deemed important to the learner. In the subsequent sections, we will consider the different combinations of teacher and oracle with certain variants of TxtEx-learning.

When defining efficiency notions for learning, the first natural notion is that of polynomial run-time: the learner must converge within $p(e)$ computation steps, where p is a polynomial and e is a code for the target. There are two problems with this definition. First, apart from trivial cases, any learning process can be delayed arbitrarily by using an enumeration that repeats a single element of the target set. Second, if a learning machine has produced an encoding of the target, but has failed to do so in polynomial run-time, a suitably larger and equivalent encoding can be chosen instead so that the run-time is appropriately bounded. As we are considering indexed families, rather than general classes of c.e. sets, we can address the second problem by fixing a reference index for every set in the family against which efficiency is measured.

Definition 2.4. Let $\mathcal{A} = \{A_n\}_{n\in\mathbb{N}}$ be an indexed family. We define the *minimal index of* $A \in \mathcal{A}$ (symbolically, $\textsc{mi}_{\mathcal{A}}(A)$) to be the least n such that $A_n = A$.

By restricting our attention to indexed families, we have a well-defined concept of polynomial bounds in the size of the target that is independent of the underlying numbering of the c.e. sets, thereby addressing the second problem. In the absence of an oracle or teacher the first problem remains. Nevertheless, we include polynomial run-time among the notions of efficiency that we define below as it is reasonable when an oracle or teacher is present.

We will address four measures of learning efficiency: Polynomial run-time, polynomial size dataset, polynomial size characteristic sample, and polynomial mind-changes.

We have also proved [3] the results presented in this paper for general classes of c.e. sets equipped with an indexing function. Nevertheless, in this paper we restrict our attention to the limited case of indexed families as it is a more familiar context than the indexed target families required by the general case. In that more general case, the indexing function selects a unique code from the underlying numbering for each set in the class. The codes output by the indexing function are taken as the reference against which efficiency is computed.

3. Polynomial Run-Time

Definition 3.1. An indexed family $\mathcal{F} = \{F_n\}_{n\in\mathbb{N}}$ is *polynomial run-time learnable* (*PRT-learnable*) if there is a machine M and a polynomial p such that for every enumeration f of $F \in \mathcal{F}$, the learner M converges to a correct index on f in fewer than $p(\textsc{mi}_{\mathcal{F}}(F))$ computation steps. If an oracle is accessed, oracle use must also be bounded by $p(\textsc{mi}_{\mathcal{F}}(F))$. We use to PRT to denote the set of all PRT-learnable indexed families.

We will apply Definition 2.3 to Definition 3.1 to obtain, for example, PRT[T]-learning and PRT[T], the PRT[T]-learnable indexed families.

Proposition 3.2 demonstrates that PRT-learnability is much too restrictive in the absence of an oracle or teacher.

Proposition 3.2. *Let $\mathcal{F}$ be an indexed family. If there are $A, B \in \mathcal{F}$ such that $A \neq B$ and $A \cap B \neq \emptyset$, then $\mathcal{F}$ is not PRT-learnable.*

Proof. Let A, B and $\mathcal{F}$ be as in the statement, let M be an arbitrary learning machine and p an arbitrary increasing polynomial. Also, let $a = \textsc{mi}_{\mathcal{F}}(A)$, $b = \textsc{mi}_{\mathcal{F}}(B)$ and let $x \in A \cap B$. Define f_A to be an enumeration of A that begins with $x^{p(a)+p(b)}$ and f_B be an enumeration of B that begins with $x^{p(a)+p(b)}$. If M PRT-identifies f_A, then $M(\sigma)$ must be a code for A for any

$\sigma = x^{p(a)+i}$ for $i \geq 0$. Similarly, if M PRT-identifies f_B, then $M(\sigma)$ must be a code for B for any $\sigma = x^{p(b)+i}$ for $i \geq 0$. Thus, no machine can PRT identify both f_A and f_B and $\mathcal{F}$ is not PRT-learnable.

□

On the other hand, there are many non-trivial indexed families which are PRT[O]-, PRT[T]- or PRT[T,O]-learnable.

Example 3.3. Define $F_n = [n,\infty)$, $G_{\langle m,n\rangle} = [m,n]$ and $H^k_n = \text{content}((n)_k)$. The indexed families $\mathcal{F} = \{F_n\}_{n\in\mathbb{N}}$, $\mathcal{G} = \{G_n\}_{n\in\mathbb{N}}$ and $\mathcal{H}_k = \{H^k_n\}_{n\in\mathbb{N}}$ for $k \in \mathbb{N}$ are PRT[O]-learnable.

Example 3.4. The indexed families in Example 3.3 are also PRT[T]-learnable. For example, consider $\mathcal{H}_k$ for some fixed k. Define a teacher T such that $T(a_0\cdots a_n) = T(a_0\cdots a_{n-1})a_n$ if $a_n \notin \{a_0,\ldots,a_{n-1}\}$ and outputs $T(a_0\cdots a_{n-1})$ otherwise. Define a learner M that waits until it has received $k+1$ distinct numbers, $\{b_0,\ldots b_k\}$, from T and then outputs $\langle b_0,\ldots b_k\rangle$. (M,T) PRT[T]-learns $\mathcal{H}_k$.

Proposition 3.5. *$PRT \subset PRT[O] \subseteq PRT[T,O]$ and $PRT \subset PRT[T] \subseteq PRT[T,O]$.*

Proof. Let $\mathcal{H}_2$ be as above. As observed in Examples 3.3 and 3.4, $\mathcal{H}_2$ is PRT[O]-learnable and PRT[T]-learnable, but by Proposition 3.2, $\mathcal{H}_2$ is not PRT-learnable. Thus, $PRT \subset PRT[O] \cap PRT[T]$. The other containments follow from the definitions.

□

We now produce indexed families that distinguish PRT[T]-learning from PRT[O]-learning and PRT[T,O]-learning from both PRT[O]- and PRT[T]-learning. In order to prove that all of these distinctions are non-trivial, we introduce the concept of marked self-description.

3.1. Marked Self-Describing Sets.

Including self-description in an object is an encoding technique on which many important learning theory examples are based. Examples of self-description include the self-describing sets $\mathcal{SD} = \{A \in \mathcal{E} : W_{min(A)} = A\}$, and the almost self-describing functions $\mathcal{ASD} = \{f : \phi_{f(0)} =^* f\}$. Many variants on the self-description theme have been explored in learning theory and inductive inference.

Our interest is in families that use carefully engineered self-description to calibrate the difficulty in identifying their members. We will construct families whose members are not only self-describing, but also have their self-describing elements marked for ease of identification. We say that such families exhibit *marked self-description*. In particular, we will use encapsulating objects that we call descriptors.

Definition 3.6. For finite $X \subset \mathbb{N}$, a *descriptor on the i^{th}-column* is a finite set $D = \{\langle x, c_x, 1, i\rangle : x \in X\} \subseteq C_i(C_1)$ such that

(1) $\sum_{x\in X} \textsc{signedInt}(c_x) = 0$
(2) $(\forall X' \subset X)\left(\sum_{x\in X'} \textsc{signedInt}(c_x) \neq 0\right)$
(3) $\sum_{x\in X'} \textsc{signedInt}(x) \geq 0$.

Such a descriptor is said to *describe* the natural number $n = \sum_{x\in X} \textsc{signedInt}(x)$. For $\langle x, c_x, 1, i\rangle \in D$, we call c_x the *completion index* of the element. For $n \in \mathbb{N}$, we define $\textsc{descriptors}_i(n)$ to be the set of all descriptors on the i^{th}-column that describe n.

A descriptor can be thought of as a stream of data that includes parity bits to check the integrity of the data stream and where the intended message is the number described by the descriptor. Thus, a machine can decide not only which elements are pieces of the descriptor (packets in the stream), but also decide when the entire descriptor has appeared in the enumeration (all the packets have been received). By using a descriptor to encode the self-description for a set, we make the self-description instantly recognizable upon appearance in the enumeration. For this reason, learning such a self-describing set can be achieved with no mind-changes. In contrast to the degree to which we have made learning easier, we have potentially made *efficient* learning harder. By distributing the self-description into a large descriptor, we will create a scenario in which a very large amount of data is required to reach a correct decision. We now proceed to our first result using these tools.

Lemma 3.7. *There is an indexed family $\{F_n\}_{n\in\mathbb{N}}$, where F_n describes n, which is PRT[T]-learnable, but not PRT[O]-learnable. We call this indexed family the* marked self-describing sets *and designate it by $\mathcal{MSD}$.*

Proof. Fix $n \in \mathbb{N}$ and let learning machine M and polynomial p be such that $n = \langle m, p^*, i\rangle$, where $\phi_m = M$ and $i \in \{0,1\}$. Without loss of generality, we may assume that p is increasing. Consider the situation where M has access to the membership oracle for the singleton $\{\langle 0,1,1,0\rangle\}$ and define a computable function q such that, for $\ell \in \mathbb{N}$, $q(\ell)$ is the greatest number about which M queries the oracle when it receives inputs which are substrings of $\langle 0,1,1,0\rangle^{\ell}$. Note that q is an increasing function. Define F_n to be a member of $\textsc{descriptors}_0(n)$ such that

$$F_n \cap [0, q(p(\langle m, p^*, 1\rangle))] = \{\langle 0,1,1,0\rangle\} \tag{1}$$

and chosen according to a fixed algorithm so that $\mathcal{MSD} = \{F_n\}_{n\in\mathbb{N}}$ is u.c.e.

First, we show that $\mathcal{MSD}$ is PRT[T]-learnable. We define a teacher T as follows. If content(σ) is not a descriptor, $T(\sigma)$ is the empty string. If $D = \text{content}(\sigma)$ describes n, then $T(\sigma) = \min(D)^i$ if $|\sigma| = |\sigma_0| + i$ where σ_0 is the shortest intial segment of σ whose content contains D and $i < n$; if $i = n$ then $T(\sigma) = \min(D)^n$. Having output $\min(D)$ n times, T proceeds by enumerating D in decreasing order. Let M be a machine that reads the output of T and returns the number of elements in the output of T. The teacher-learner pair learns $\mathcal{MSD}$ and the run-time of the learner is linear in the index of the target.

We now show that $\mathcal{MSD}$ is not PRT[O]-learnable. To prove that $\mathcal{MSD}$ is not PRT[O]-learnable, fix a learner $M = \phi_m$, an increasing polynomial p encoded by p^*, $n_0 = \langle m, p^*, 0\rangle$ and $n_1 = \langle m, p^*, 1\rangle$. If M PRT[O]-learns $\mathcal{MSD}$ with polynomial bound p, then it must succeed at identifying F_{n_0} and F_{n_1} within $p(n_1) \geq p(n_0)$ computation stages. Choose T_0 and T_1 to be any enumerations of F_{n_0} and F_{n_1}, respectively, which have $\langle 0,1,1,0\rangle^{p(n_1)}$ as an initial segment. When trying to identify T_0 and T_1, the learner must reach its final hypothesis before finding any elements of the target sets, F_{n_0} and F_{n_1}, other than $\langle 0,1,1,0\rangle$. Whatever hypothesis M converges to before completing the $p(n_1)$ length initial segment of either enumeration cannot code both sets. Thus, M fails to learn at least one of the two sets. Since M and p were chosen arbitrarily we conclude that $\mathcal{MSD}$ is not PRT[O]-learnable.

□

Lemma 3.8. *If $\mathcal{F} = \{F_n\}_{n\in\mathbb{N}}$ is an indexed family, p a polynomial and there are indices $a, b_0, b_1, \dots b_{p(a)-1}$ such that $F_{b_0} \subset F_{b_1} \subset \ldots \subset F_{p(a)-1} \subset F_a$, then $\mathcal{F}$ is not PRT[T]-learnable with polynomial bound p.*

Proof. Let $\mathcal{F}$, p, a and $b_0, \dots, b_{p(a)-1}$ be as in the statement and let (M, T) be an arbitrary learner-teacher pair. Let σ_0 be an initial segment of an enumeration of F_{b_0} on which $M \circ T$ outputs an index for F_{b_0} (if no such σ_0 exists, then (M, T) has already failed to learn $\mathcal{F}$). Given σ_n, an initial segment of an enumeration of F_{b_n} for $n < p(a) - 1$, define σ_{n+1} to be an initial segment of an enumeration of $F_{b_{n+1}}$ extending σ_n on which (M, T) outputs an index for $F_{b_{n+1}}$. Again, if no such extension can be found, then M has failed to learn $\mathcal{F}$. Let T be an enumeration of F_a which has $\sigma_{p(a)-1}$ as an initial segment. Since M changes hypothesis at least $p(a)$ times on $\sigma_{p(a)-1}$, either M fails to identify T or the runtime of the learner cannot be bounded by $p(a)$.

□

Lemma 3.9. *There is a PRT[O]-learnable indexed family that is not PRT[T]-learnable. We call this indexed family the* column self-describing sets *and designate it by $\mathcal{CSD}$.*

Proof. Define

$$a_n = n + 1 + \sum_{i=0}^{n-1} p_i(a_i), \tag{2}$$

where p_i is the polynomial such that $p_i^* = i$. Fix $n \in \mathbb{N}$ and define $A_n = [0, a_n] \otimes \{p_n(a_n)\} \cup \bigcup_{i<p(a_n)} [0, a_n + i] \otimes \{i\}$ and $B_{n,i} = \bigcup_{j \le i} [0, a_n + j] \otimes \{j\}$, for $i < p(a_n)$. Finally, define $F_n = A_i$ if $n = a_i$ and $F_n = B_{i,j}$ if $n = a_i + j$ where $j < p_i(a_i)$. Let $\mathcal{CSD} = \{F_n\}_{n \in \mathbb{N}}$. To PRT[O]-learn $\mathcal{CSD}$, define M to be a learning machine that uses the exponential query search algorithm to find the highest index non-empty column, queries about the members of the column, in increasing order, until the greatest element is found, and returns the value of this element. Since the number of queries involved is polynomially bounded in e, M witnesses the desired learnability.

Since $B_{n,0} \subset B_{n,1} \subset \cdots \subset B_{n,p(a_n)-1} \subset A_n$, for each polynomial, p, there is a subfamily of $\mathcal{F}$ that cannot be PRT[T]-learned with efficiency bound p. Thus, $\mathcal{F}$ is not PRT[T]-learnable.

□

Finally, we wish to distinguish PRT[T,O]-learning from both PRT[T]-learning and PRT[O]-learning.

Lemma 3.10. *There is an indexed family which is PRT[T,O]-learnable, but neither PRT[T]-learnable nor PRT[O]-learnable.*

Proof. To prove the claim, we must combine the strategies used in the proofs of Lemma 3.7 and Lemma 3.9. Define F_n exactly as the members of $\mathcal{MSD}$ are defined except we modify formula (1) to be

$$F_n \cap [0, q(p(3\langle m, p^*, 1\rangle))] = \{\langle 0, 1, 1, 0\rangle\}.$$

We also define G_n exactly as the members of $\mathcal{CSD}$ are defined except that we replace formula (2) by

$$a_n = 3(n+1) + \sum_{i=0}^{n-1} p_i(3a_i)$$

Finally, we define $\mathcal{H} = \{H_n\}_{n\in\mathbb{N}}$ where

$$H_n = \begin{cases} G_i & \text{if } n = 2i \\ F_i & \text{if } n = 2i+1 \end{cases}.$$

We will show that $\mathcal{H} = \{H_n\}_{n\in\mathbb{N}}$ is PRT[T,O]-learnable, but neither PRT[T]-learnable nor PRT[O]-learnable. To PRT[T,O]-learn $\mathcal{H}$, let M be a learner which first determines if the target set contains 0 using an oracle query. If the target does, then M proceeds as the PRT[O]-learner in the proof of Lemma 3.9, multiplying the hypotheses output by that learner by 2. If the target does not contain 0, then M proceeds as the PRT[T]-learner in the proof of Lemma 3.7, multiplying the hypotheses output by that learner by 2 and adding 1. M PRT[T,O]-learns $\mathcal{H}$ with only a linear decrease in efficiency compared to the two learners from the previous Lemmas.

To see that $\mathcal{H}$ is neither PRT[O]-learnable nor PRT[T]-learnable, observe that the proofs of Lemmas 3.7 and 3.9 suffice to show that $\mathcal{H}$ contains two indexed subfamilies, one of which fails to be PRT[O]-learnable and the other fails to be PRT[T]-learnable.

□

For clarity, we summarize the results of Section 3 in the following theorem.

Theorem 3.11.

(1) $PRT \subset PRT[O] \subset PRT[T,O]$,
(2) $PRT \subset PRT[T] \subset PRT[T,O]$,
(3) $PRT[O] \setminus PRT[T] \neq \emptyset$,
(4) $PRT[T] \setminus PRT[O] \neq \emptyset$.

Proof. All of the claims in the statement follow from Lemmas 3.7, 3.9 and 3.10 and Proposition 3.5.

□

4. Polynomial Size Dataset

Definition 4.1. An indexed family, $\mathcal{F} = \{F_n\}_{n\in\mathbb{N}}$, is *polynomial size dataset learnable* (*PSD-learnable*) if there is a machine M and a polynomial p such that for any enumeration f of $F \in \mathcal{F}$, M converges to a correct index on an initial segment $f \upharpoonright n$ such that $|\{f(x) : x < n\}| < p(\text{MI}_{\mathcal{F}}(F))$. If an oracle is accessed, oracle use must also be bounded by $p(\text{MI}_{\mathcal{F}}(F))$.

Note that oracle use bounds both the queries to which the oracle reponds in the positive and those to which it responds in the negative. We shall apply Definition 2.3 to Definition 4.1 much as we did in the case of Definition 3.1.

Proposition 4.2. *$PSD \subseteq PSD[O] \subseteq PSD[T,O]$ and $PSD \subseteq PSD[T] \subseteq PSD[T,O]$.*

Proof. The claim follows from the definitions of PSD, PSD[T], PSD[O] and PSD[T,O].

□

Unlike PRT-learning, there are non-trivial PSD-learnable indexed families.

Example 4.3. Let $\mathcal{F}$ be an indexed family containing all the finite sets such that $\text{MI}_{\mathcal{F}}(F) = \langle |F|, e\rangle$, where e is the canonical code for F. $\mathcal{F}$ is PSD-learnable by the learning machine M where $M(a_0 \cdots a_n) = \langle |\text{content}(a_0 \cdots a_n)|, a\rangle$, where a is the canonical code content$(a_0 \cdots a_n)$.

Example 4.4. Let $F_n = [0, 2^n]$. $\mathcal{F} = \{F_n\}_{n\in\mathbb{N}}$ is PSD[O]-learnable by a learning machine that uses the exponential query search algorithm to find the greatest element. $\mathcal{F}$ is PSD[T]-learnable by the pair (M, T) where $M(a_0^{k_0}, \ldots, a_{n-1}^{k_{n-1}}) = k_0$ and $T(a_0 \cdots a_{k+1}) = a_0^n a_1 \ldots, a_{k+1}$ if $2^n \leq \max\{a_0, \ldots, a_{k+1}\} < 2^{n+1}$ and $\max\{a_0, \ldots, a_k\} < 2^n$. $\mathcal{F}$ is not PSD-learnable as the learner may be forced to receive 2^{n-1} distinct elements before converging to a correct hypothesis.

Lemma 4.5. *There is an indexed family which is PSD[T]-learnable, but not PSD[O]-learnable.*

Proof. We prove the claim using a strategy similar to that used in the proof of Lemma 3.7. Following the notation established in the proof, the only differences are that we define $q(\ell)$ to be the maximum number about which M queries the oracle when it receives inputs which are substrings $\langle 0,1,1,0\rangle\langle 2,1,1,0\rangle\cdots\langle 2\ell,1,1,0\rangle$, given the oracle for $\{\langle 2i,1,1,0\rangle : i \leq \ell\}$, and that we define F_n to be a member of $\textsc{descriptors}_0(n)$ such that

$$F_n \cap [0, q(p(\langle m, p^*, 1\rangle))] = \{\langle 2i, 1, 1, 0\rangle : i \leq p(\langle m, p^*, 1\rangle)\}.$$

Let $\mathcal{F} = \{F_n\}_{n\in\mathbb{N}}$. The proof that $\mathcal{F}$ is PSD[T]-learnable is exactly the same as the proof that $\mathcal{MSD}$ is PRT[T]-learnable. That $\mathcal{F}$ is not PSD[O]-learnable follows from the observation that for abitrary $M = \phi_m$, M cannot distinguish between $F_{\langle m,p^*,0\rangle}$ and $F_{\langle m,p^*,1\rangle}$ on increasing enumerations without receiving more than $p(\langle m, p^*, 1\rangle)$ elements of an enumeration of the target.

□

Theorem 4.6.

(1) PSD ⊂ PSD[O] ⊂ PSD[T,O],
(2) PSD ⊂ PSD[T] ⊂ PSD[T,O],
(3) PSD[O] \ PSD[T] ≠ ∅,
(4) PSD[T] \ PSD[O] ≠ ∅.

Proof. By Lemma 4.5 and Example 4.4, we need only prove that PSD[O] \ PSD[T] ≠ ∅ and PSD[T] ∪ PSD[O] ⊂ PSD[T,O].

Observe that the proof of Lemma 3.8 demonstrates that an indexed family meeting the hypotheses of the lemma is not PSD[T]-learnable. Thus, Lemma 3.9 proves that $\mathcal{CSD} \in$ PSD[O] \ PSD[T].

Following the proof of Lemma 3.10, merging the families constructed in Lemmas 4.5 and 3.9 with suitable modifications produces an indexed family which is PSD[T,O]-learnable, but neither PSD[T]-learnable nor PSD[O]-learnable.

□

It follows from the definitions that PRT ⊆ PSD, PRT[O] ⊆ PSD[O], PRT[T] ⊆ PSD[T] and PRT[T,O] ⊆ PSD[T,O]. With the following theorem, we show that all three of these containments are strict.

Theorem 4.7. *PSD \ PRT[T, O] ≠ ∅*

Proof. Let K denote the halting problem. Define $\mathcal{F} = \{F_0, F_1, \ldots\}$ where

- $F_{2i+1} = \{2i\}$ if $i \notin K$ and $F_{2i+1} = \{2i, 2i+1\}$ if $i \in K$.
- $F_{2^{2^i}} = \{2i, 2i+1\}$.
- For all even numbers $2i \neq 2^{2^k}$ for some k, $F_{2i} = \emptyset$.

That $\mathcal{F}$ is PSD-learnable is witnessed by the learner M which outputs 6 on the empty string, outputs $2i+1$ on a string with one unique element which is $2i$ and outputs 2^{2^i} and any other string, if the string either contains $2i$ or $2i+1$. On the other hand, suppose that the learner-teacher pair, (N,T), PRT[T]-learns $\mathcal{F}$ with polynomial bound p. Computing and returning 2^{2^i} cannot be done within $p(2i+1)$ computation steps for more than finitely many values of i; thus, for all but finitely many i, $i \notin K$ if and only if $(\exists j)(N(T(2i\ (2i+1)^j)) = 2^{2^i})$. Since this would imply that $\overline{K}$ is Σ^0_1, we have arrived at a contradiction and must conclude that $\mathcal{F} \notin \text{PRT}[T]$. Observe that the use of an oracle does not facilitate learning in this case and so we conclude that PSD \ PRT[T,O] $\neq \emptyset$.

□

5. Polynomial Mind Changes

Definition 5.1. An indexed family, $\mathcal{F} = \{F_n\}_{n\in\mathbb{N}}$, is *polynomial mind-changes learnable* (*PMC-learnable*) if there is a machine M and a polynomial p such that for every enumeration f of $F \in \mathcal{F}$, the hypothesis stream, g, generated by M on f satisfies $|\{i : g(i) \neq g(i+1)\}| \leq p(\text{MI}_{\mathcal{F}}(F))$ and the only one that appears infinitely many times in g is an index of F. If an oracle is accessed, oracle use must also be bounded by $p(\text{MI}_{\mathcal{F}}(F))$.

We begin with an example exhibiting three PMC-learnable indexed families.

Example 5.2. Let $\mathcal{F}$ be an indexed family containing all the finite sets such that $\text{MI}_{\mathcal{F}}(F) = \langle |F|, e\rangle$, where e is the canonical code for F. $\mathcal{F}$ is PMC-learnable as witnessed by the learning machine M such that $M(a_0 \cdots a_k) = \langle |\text{content}(a_0,\ldots,a_k)|, e\rangle$, where e is the canonical code for the finite set of distinct elements in $a_0,\ldots,a_k$. On any enumeration of a finite set, F, M will change its hypothesis at most $|F|$ times.

Let $\mathcal{F} = \{F_n\}_{n\in\mathbb{N}}$, where $F_n = [0,2^n]$. $\mathcal{F}$ is PMC-learnable. Define M such that $M(\sigma)$ is a code for $[0,2^s]$, where s is the least integer greater than or equal to $\log_2(\max(\sigma))$.

$\mathcal{MSD}$ is PMC-learned by a learning machine that waits until a descriptor has appeared in the enumeration and then outputs the number the descriptor describes.

Theorem 5.3. *PMC[T] = PMC = PSD[T] and PMC[T,O] = PMC[O].*

Proof. Fix an arbitrary indexed family $\mathcal{F}$. If (M,T) PMC[T]-learns $\mathcal{F}$, then $M \circ T$ PMC-learns $\mathcal{F}$. Since every PMC-learnable indexed family is also PMC[T]-learnable, PMC = PMC[T]. Similarly, PMC[T,O] = PMC[O].

Suppose (M,T) PSD[T]-learns $\mathcal{F}$ and define M^* such that $M^*(a_0\cdots a_{k+1}) = M \circ T(a_0 \cdots a_{k+1})$ when $T(a_0\cdots a_{k+1}) \neq T(a_0\cdots a_k)$ and $M^*(a_0\cdots a_{k+1}) = M^*(a_0\cdots a_k)$, otherwise. Since (M,T) PSD[T]-learns $\mathcal{F}$, the number of distinct elements that T outputs before $M\circ T$ converges to a correct hypothesis is polynomially bounded, hence M^* changes hypothesis a polynomially bounded number of times. Thus, $\mathcal{F} \in \text{PMC}$

Define functions f and g such that $f(\sigma) = |\sigma|$ and $g(n,x) = x^n$, the string x repeated n times. Suppose M PMC-learns $\mathcal{F}$. Define T such that $T(\sigma) = g(M(\sigma), \min(\sigma))$ if $M(\sigma)$ is different from $M(\tau)$ for all $\tau \prec \sigma$. $T(\sigma)$ is undefined otherwise. (f,T) PSD[T]-learns $\mathcal{F}$ because it converges to a correct hypothesis after reading a polynomially bounded number of outputs from T.

□

Theorem 5.4. *PMC = PMC[T] ⊂ PMC[O] = PMC[T,O]*

Proof. The proof of Lemma 3.8 implies that indexed families which meet the hypotheses are not PMC-learnable. Thus, $\mathcal{CSD}$ is PMC[O]-learnable, but not PMC-learnable. By Theorem 5.3, PMC = PMC[T] and PMC[T] = PMC[T,O]. Hence, the desired claims are true. □

6. Polynomial Size Characteristic Sample

Definition 6.1. An indexed family, $\mathcal{F}$, is *polynomial size characteristic sample learnable* (*PCS-learnable*) if there is a machine M, a polynomial p and a family $\mathcal{H}$ such that for each $F \in \mathcal{F}$, there is a corresponding $H \in \mathcal{H}$ such that $|H| < p(\textsc{mi}_{\mathcal{F}}(F))$ and if f is an enumeration of F, then M outputs the same encoding of F on every initial segment of f whose content includes H. If an oracle is accessed, oracle use must also be bounded by $p(\textsc{mi}_{\mathcal{F}}(F))$.

Theorem 6.2.

(1) PCS[O] \ PCS[T] ≠ ∅,
(2) PCS[T] \ PCS[O] ≠ ∅ and
(3) PCS[T,O] \ (PCS[T] ∪ PCS[O]) ≠ ∅.

Proof. Define $G_0 = \mathbb{N}$, $G_n = [0,n]$ for $n > 0$, and $\mathcal{G} = \{G_n\}_{n\in\mathbb{N}}$. $\mathcal{F}$ is PCS[O]-learned by M, where $M(a_0 \cdots a_n) = 0$ if the answer to a query about $\max\{a_0,\ldots,a_n\}+1$ is TRUE and is a code for $[0,\max\{a_0,\ldots,a_n\}]$ otherwise. Since any string, σ, can either be extended to an enumeration of $G_0 = \mathbb{N}$ or to an enumeration of $G_n = [0,n]$ for any $n \geq \max(\text{content}(\sigma))$, no learner-teacher pair can PCS-learn $\mathcal{G}$. Thus, we have proved 1.

Fix k and suppose that $k = \langle n, p^* \rangle$, where p is an increasing polynomial, and let M be the learner coded by n. Define E_k to be a c.e. subset of $[2^{2k+1}+1, 2^{2k+2}]$ that satisfies three conditions.

- $2^{2k+1}+1 \in E_k$.
- $|E_k| = p(2k+1)+1$.
- For any enumeration, f, of $[2^{2k+1}+1, 2^{2k+2}]$, if $E_k \subset f \upharpoonright i$ for some $i \leq p(2k+1)$, then $M(f \upharpoonright j) = 2k$ for $i \leq j \leq p(2k+1)$.

If no such set exists, let $E_k = \emptyset$. If $E_k \neq \emptyset$, we define a set D_k satisfiying the following conditions.

- $|D_k| = 2p(2k+1)+1$.
- $E_k \subset D_k \subset [2^{2k+1}+1, 2^{2k+2}]$.
- D_k includes the first $p(2k+1)$ members of $[2^{2k+1}+1, 2^{2k+2}]$ about which M queries the oracle on a fixed uniformly computable enumeration of E_k.

If $E_k = \emptyset$, then $D_k = \emptyset$. To prove 2, define $\mathcal{F} = \{F_0, F_1, \ldots\}$ where $F_{2k} = [2^{2k+1}+1, 2^{2k+2}]$ and $F_{2k+1} = D_k \cup \{2^{2k+1}+1\}$. Observe that for any oracle learner, M, and polynomial, p, there is a k such that either

- there is an enumeration of F_{2k+1} on which M converges to $2k$ or M makes more than $p(2k+1)$ oracle queries, or
- M does not have a characteristic sample for F_{2k} of size at most $p(2k)$.

Thus, $\mathcal{F}$ is not PCS[O]-learnable. On the other hand, consider the learner, M, and teacher, T, defined as follows. Once a number of the form $2^{2k+1}+1$ appears in the enumeration, T

outputs $2^{2k+1}+1$. Using k, T then determines a natural number, n, and polynomial, p, such that $k = \langle n, p\rangle$. The teacher outputs no further numbers until the distinct elements of the enumeration exceeds $2p(2k+1)+1$. At this point, T outputs $2^{2k+1}+2$. Simultaneously, T calculates D_k. If D_k is nonempty, then T outputs the least element of $[2^{2k+1}+1, 2^{2k+2}] \setminus D_k$ if it appears in the enumeration. M returns $2k+1$ if T has output only one element and returns $2k$ if T has output two or more elements. The learner-teacher pair PCS[T]-learns $\mathcal{F}$, proving 2.

We prove 3 by combining the two families defined above into one family: define $\mathcal{H} = \{G_0 \otimes \{0\}, F_0 \otimes \{1\}, G_1 \otimes \{0\}, F_1 \otimes \{1\}, \ldots\}$. Were $\mathcal{H}$ PCS[O]-learnable, that would imply that $\mathcal{F}$ is PCS[O]-learnable; similarly, if $\mathcal{H}$ were PCS[T]-learnable then $\mathcal{G}$ would also be PCS[T]-learnable. That $\mathcal{H}$ is PCS[T,O]-learnable is witnessed by a learner-teacher pair (with access to an oracle) that first waits to see whether the enumeration contains elements of the form $\langle n, 0\rangle$ or $\langle n, 1\rangle$ and applies the appropriate learning algorithm as defined above.

□

The final theorem of this paper illustrates some of the relationships between PCS-learning and the other three types of polynomial-bounded learning.

Theorem 6.3.

(1) PMC $\setminus$ PCS $\neq \emptyset$.
(2) PSD $\subset$ PCS.
(3) PCS $\setminus$ PMC $\neq \emptyset$.

Proof. Observe that the family, $\mathcal{F}$, defined in the proof of Theorem 6.2 is also PMC-learnable. Consider a learner, M, which attempts to compute D_k and returns 0 until a number of the form $2^{2k+1}+1$ appears in the enumeration. If this is the only number in the enumeration, then M outputs $2k+1$. M also outputs $2k+1$ if M succeeds in computing D_k and every element of the enumeration is in $D_k \cup \{2^{2k+1}+1\}$. Otherwise, M outputs $2k$. Since $\mathcal{F} \notin$ PCS, we have proved 1.

We prove 2 in two parts. First, suppose that M PSD-learns a family $\mathcal{G} = \{G_0, G_1, \ldots\}$ with polynomial bound p. Let C_i denote the first at most $p(i)$ elements of G_i. Define M^* such that $M^*(\sigma) = M(\tau)$, where τ lists the distinct elements of σ in increasing order. Since M must PSD-learn G_i on the increasing enumeration, C_i must be a characteristic sample for M^* on G_i. Thus, PSD $\subseteq$ PCS. Now, consider $\mathcal{A} = \{A_0, A_1, \ldots\}$ where $A_n = \{n\} \oplus \mathbb{N}$. Consider the string α_k consisting of the odd numbers from 1 to $2k+1$. For any member of $\mathcal{A}$, there is an enumeration that begins with α_k. Consequently, $\mathcal{A}$ is not PSD-learnable. Conversely, each member of $\mathcal{A}$ has a characteristic sample of size 1. We conclude that PSD $\subset$ PCS.

Given $n \in \mathbb{N}$, there are unique i_n and k_n such that $n = i_n + 2^{k_n}$ and $1 \leq i_n \leq 2^{k_n}$. Define $\mathcal{G} = \{G_0, G_1, \ldots\}$ where $G_{2n} = \{n\} \oplus [0, 2^n]$ and $G_{2n+1} = \{k_n\} \oplus [0, i_n]$. In order to PCS-learn $\mathcal{F}$, we define a learner M as follows. Let σ be an arbitrary string of natural numbers. If σ contains no odd numbers or contains no even numbers, define $M(\sigma) = 0$. Otherwise, let $2n$ be the least even number in σ and let $2m+1$ be the greatest odd number. $M(\sigma) = 2n$ if $m = 2^n$ and $M(\sigma) = 2k+1$, where $k = m + 2^n$, if $m \neq 2^n$. Each member of $\mathcal{G}$ has a characteristic sample of size 2 for M, thus, M PCS-learns $\mathcal{G}$. Conversely, suppose that N PMC-learns $\mathcal{G}$. For each $n, i, a_1, a_2, \ldots, a_{i-1}$ and $k = i + 2^n$, there is an a_i such that $N(2n\ 1\ 3^{a_1}\ 5^{a_2} \ldots (2i-1)^{a_{i-1}}\ (2i+$

$1)^{a_i}) = 2k+1$. Thus, for any polynomial there is an n such that $p(2n) < 2^n$ and an enumeration of G_{2n} on which N outputs 2^n different hypotheses. We have proved 3.

□

7. Acknowledgements

The authors could like to thank the anonymous referee for numerous useful comments and suggestions.

References

[1] Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.
[2] Dana Angluin. Queries revisited. *Theor. Comput. Sci.*, 313(2):175–194, February 2004.
[3] Achilles Beros and Achilles de la Higuera. Teachers, learners and oracles. http://arxiv.org/abs/1504.03623.
[4] Thorsten Doliwa, Hans Ulrich Simon, and Sandra Zilles. Recursive teaching dimension, learning complexity, and maximum classes. In *Proceedings of the 21st International Conference on Algorithmic Learning Theory (ALT'10)*, pages 209–223. Springer, 2010.
[5] Sally Floyd and Manfred Warmuth. Sample compression, learnability, and the vapnik-chervonenkis dimension. *Machine Learning*, 21:1–36, 1995.
[6] Lance Fortnow, William Gasarch, Sanjay Jain, Efim Kinber, Martin Kummer, Stuart Kurtz, Mark Pleszkovich, Theodore Slaman, Robert Solovay, and Frank Stephan. Extremes in the degrees of inferability. *Annals of Pure and Applied Logic*, 66(3):231 – 276, 1994.
[7] Sally A. Goldman and Michael J. Kearns. On the complexity of teaching. *J. Comput. Syst. Sci.*, 50(1):20–31, February 1995.
[8] Sanjay Jain and Arun Sharma. On the non-existence of maximal inference degrees for language identification. *Information Processing Letters*, 47(2):81 – 88, 1993.
[9] Martin Kummer and Frank Stephan. On the structure of degrees of inferability. *Journal of Computer and System Sciences*, 52(2):214 – 238, 1996.
[10] Claude Sammut and Ranan Banerji. Learning concepts by asking questions.
[11] Frank Stephan. Noisy inference and oracles. *Theoretical Computer Science*, 185:129–157, 1997.
[12] Sandra Zilles, Steffen Lange, Robert C. Holte, and Martin Zinkevich. Teaching dimensions based on cooperative learning. In *Proceedings of the 21st Annual Conference on Learning Theory (COLT'08)*, pages 135–146, 2008.

(Achilles A. Beros) Department of Mathematics, University of Hawai'i at Mānoa, Honolulu, HI 96822, USA
E-mail address: `beros@math.hawaii.edu`

(Achilles A. Beros) Laboratoire d'Informatique de Nantes Atlantique, Université de Nantes, 2 rue de la Houssinière BP 92208, 44322 Nantes Cedex 03, FRANCE
E-mail address: `cdlh@univ-nantes.fr`